ICCAD 2015 Contest

Color Balancing for Double Patterning

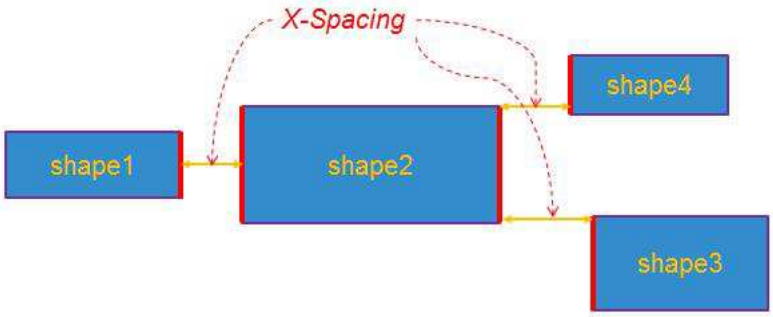
Ru-Lin Yang, Vincent Hsu
Synopsys Taiwan Co., Ltd. Hsinchu Science Park Office

Introduction

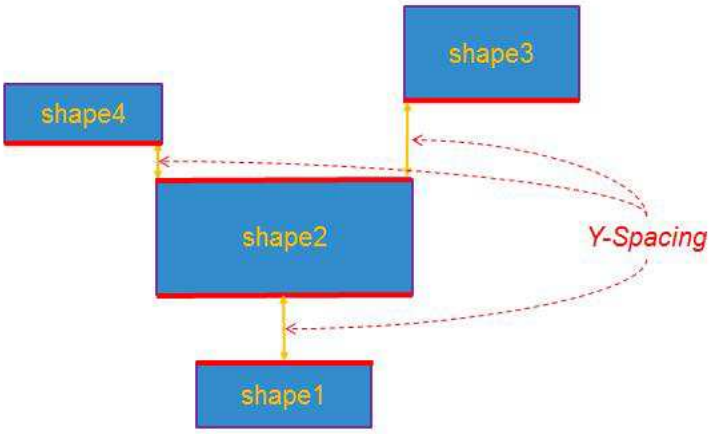
The traditional lithography using 193 nm wavelength light cannot print patterns at advanced nodes. Therefore, foundry provides a scheme employing extra masks (multiple patterning technology) to make existing lithography work at these advanced nodes. Double patterning technology (DPT) is a method of breaking up a layout so that sub-resolution patterns are separated onto two distinct masks. These two masks are exposed and processed sequentially to create the original design patterns by composing the layout features obtained from the independent patterning steps [1]. Typically, the problem of separating layout patterns onto two masks is solved by transforming it into a 2-coloring problem [2]. Patterns being assigned the same color will be on the same mask. Non-uniform color (pattern) density on a mask can cause more pattern distortion and create more hotspots [3], while a balanced coloring would allow more space for scattering bar insertion during OPC (optical proximity correction) that leads to better patterning quality. Thus, balanced and uniform color density is preferred during layout decomposition.

Terminology

1. Shape: A shape is a rectangle which is also a coloring unit in layout. This implies that a shape can only be assigned at most one color. If a shape has a color, it is called a *colored shape*. Otherwise, it is called a *non-colored shape*.
2. *X-Spacing*: If a vertical edge of a shape can see a vertical edge of another shape in X-coordinate direction, the distance between the two vertical edges is *X-Spacing*. If *X-Spacing* is < α, the two shapes must have different colors. Here, α is called *minimum X-Spacing*.



3. *Y-Spacing*: If a horizontal edge of a shape can see a horizontal edge of another shape in Y-coordinate direction, the distance between the two horizontal edges is *Y-Spacing*. If *Y-Spacing* is < β, the two shapes must have different colors. Here, β is called *minimum Y-Spacing*.



4. Coloring Graph: A coloring graph is a connected graph where a vertex represents a shape and an edge is created between two vertices (shapes) whose *X-Spacing* is < α or *Y-Spacing is* < β. That is, the two shapes must be assigned different colors.
5. Color Conflict: If there exists an odd cycle in a coloring graph, the coloring graph is not 2-colorable. None of the vertices in a non-2-colorable graph will be assigned a color.

6. Coloring Bounding Box: A smallest bounding box that contains all colored shapes. Note that there is only one coloring bounding box for a given layout design. A coloring bounding box may contain non-colored shapes.
7. Color Density Window: It is a square inside a coloring bounding box.
8. Color Density Window Size (ω): It is the width and height of a color density window.
9. Color Density: it is defined as $\frac{C}{U} * 100$ where *C* is the total area of colored shapes of the same color in a color density window (excluding the portion outside the color density window) and *U* is the area of the color density window. Keep two digits below the floating point and use rounding to keep hundredth precision. If a colored shape is not completely inside a color density window, only the area contained in the color density window is counted when color density is calculated.

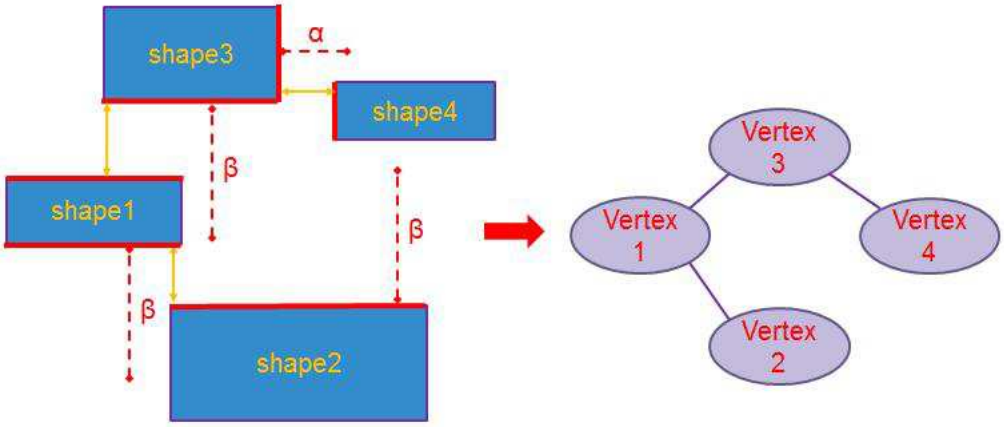
Problem Specification

Given a layout design which contains only non-abutting and non-overlapping shapes, minimum X-Spacing α, minimum Y-Spacing β, and color density window size ω, color all shapes in the design using 2 colors, *color-A* and *color-B*. If a coloring graph is not 2-colorable, all of its vertices must not be colored. The objective is to minimize the difference between color-A density and color-B density as much as possible for all color density windows.

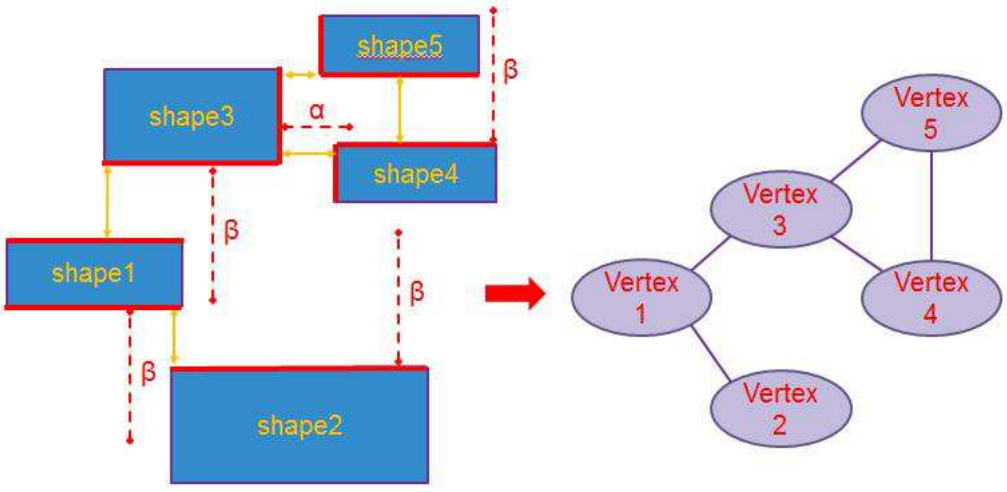
Build Coloring Graph

Given any two shapes, if they satisfy (*X-Spacing* < α) or (*Y-Spacing*< β), generate an edge connecting the two vertices corresponding to these two shapes. Two examples are illustrated as follows.

Example One: Because shape3 and shape1 have their *Y-Spacing* < β, we generate an edge connecting Vertex 3 and Vertex 1. Similarly, we generate an edge connecting Vertex 1 and Vertex 2. Because shape3 and shape4 have their *X-Spacing* < α, we generate an edge connecting Vertex 3 and Vertex 4.

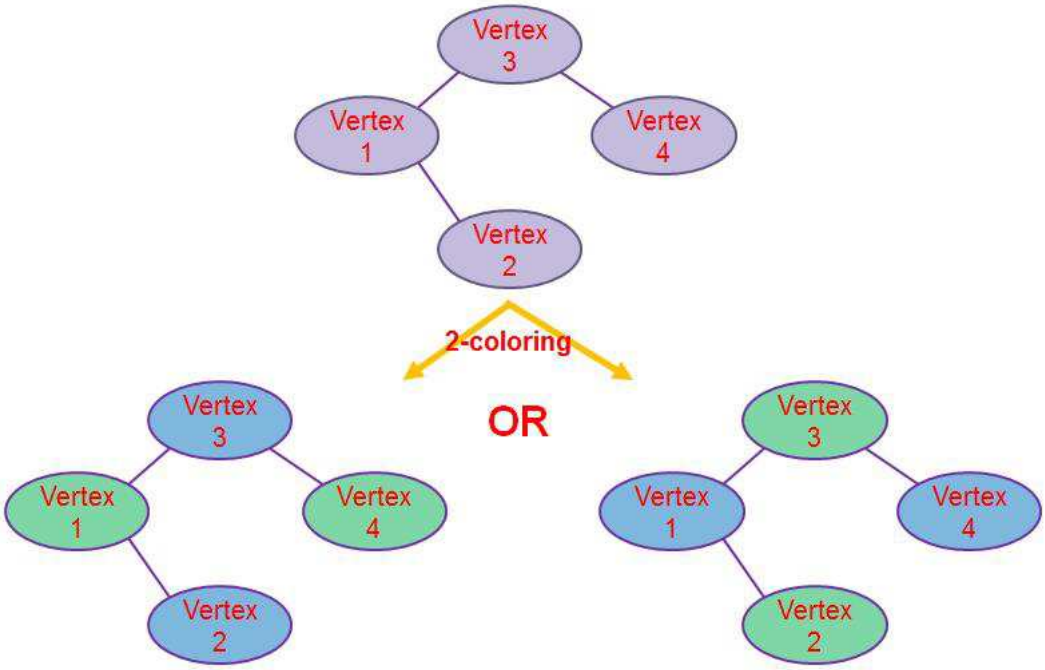


Example Two: Because shape3 and shape1 have their *Y-Spacing* < β, we generate an edge connecting Vertex 3 and Vertex 1. Similarly, we generate an edge connecting Vertex 1 and Vertex 2 and an edge connecting Vertex 5 and Vertex 4 . Because shape3 and shape4 have their *X-Spacing* < α, we generate an edge connecting Vertex 3 and Vertex 4. Similarly, we generate an edge connecting Vertex 3 and Vertex 5.

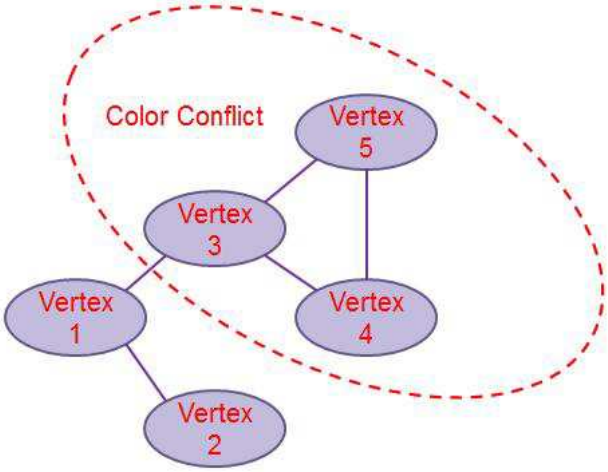


Coloring

Apply a 2-coloring scheme to coloring graphs. Any two adjacent vertices must have different colors. If one has color-A, the other must have color-B. An illustration is as follows.



If a coloring graph is not 2-colorable, do not assign any color to its vertices. An illustration is as follows. There are no colors on Vertex 1, Vertex 2, Vertex 3, Vertex 4, and Vertex 5 that correspond to the five shapes.

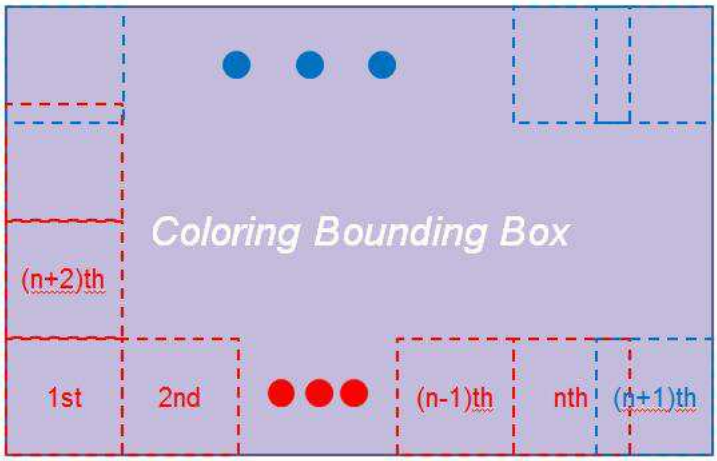


Calculating Color Density

Generate the color density windows to totally cover a coloring bounding box based on the following guidelines.

1. The first color density window is placed at the bottom-left corner of the coloring bounding box.
2. The second color density window is placed at the right hand side of the first color density window.
3. The next color density window is placed at the right hand side of the current color density window. If the new color density window overlaps the right edge of the coloring bounding box, the window is shifted left until it abuts the right edge of the coloring bounding box.
4. The next color density window is placed at the left-most position of the next row. The next row is on the top of the current row.
5. If the new color density window overlaps the top edge of the coloring bounding box, the window is shifted down until it abuts the top edge of the coloring bounding box.

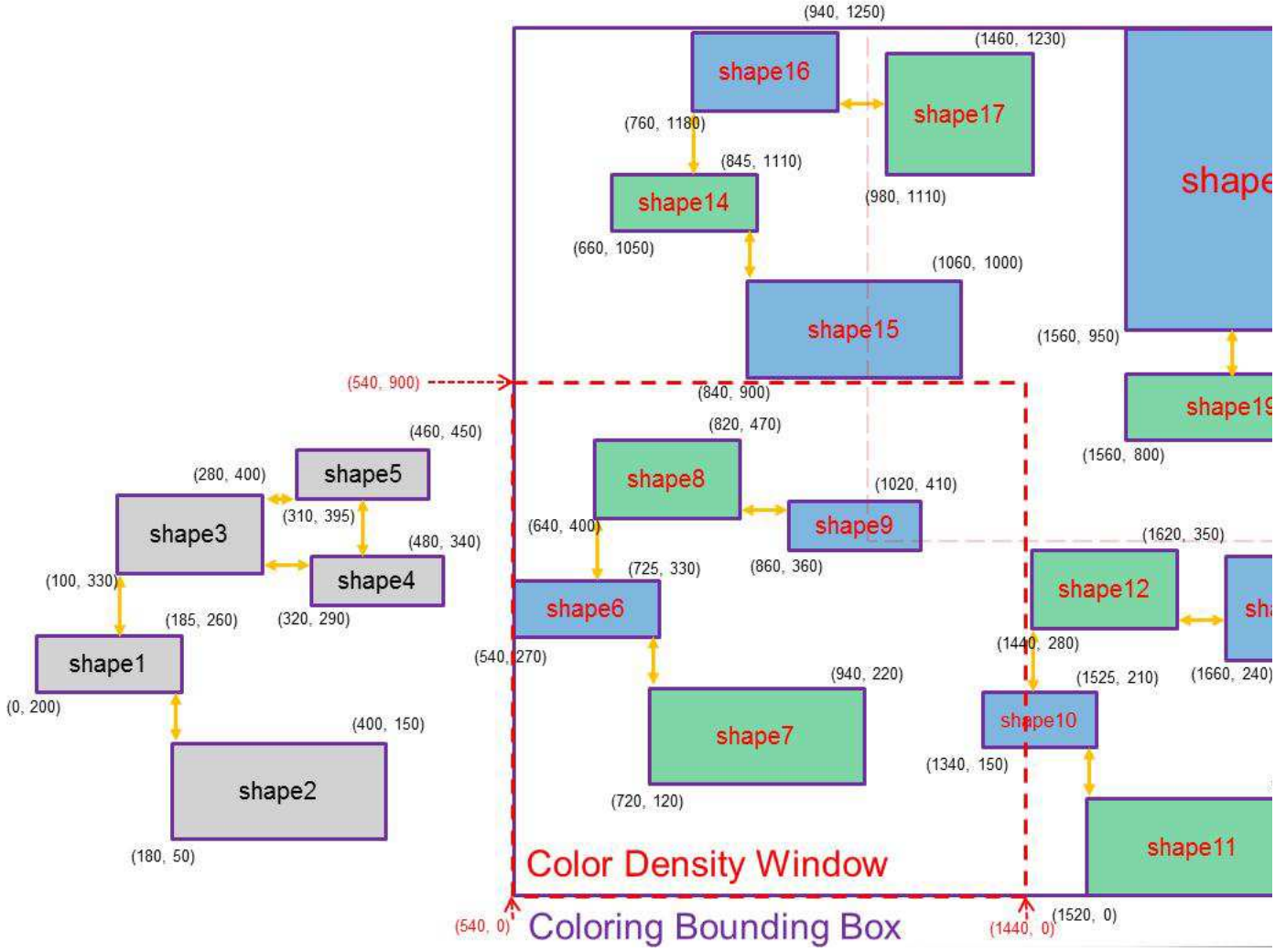
An illustration is as follows.



Given a color density window, its color density is calculated as follows.

1. The area of color-A: Sum the area of the shapes having color-A in the color density window.
2. The area of color-B: Sum the area of the shapes having color-B in the color density window.
3. Calculate color-A density based on the color density definition.
4. Calculate color-B density based on the color density definition.

An illustration is as follows.



The area of the color density window = 900×900 = 810000

The area of color-A in the color density window = (940-720)(220-120)+(820-640)(470-400) = 34600

The area of color-B in the color density window = (725-540)(330-270)+(1020-860)(410-360)+(1440-1340)(210-150) = 25100

The color-A density = 34600÷810000 = 0.04271… ≈ 4.27%

The color-B density = 25100÷810000 = 0.03098… ≈ 3.10%

Input Format

The input file is an ASCII text file giving α, β, ω, and the coordinates of rectangles. There is a comma (,) between two consecutive coordinates. Below is an input file for a design with *m* rectangles.

ALPHA=α

BETA=β

OMEGA=ω

$x1_1, y1_1, x1_2, y1_2$

…

xi_1, yi_1, xi_2, yi_2

…

xm_1, ym_1, xm_2, ym_2

For i=1 to m, xi_1 and yi_1 are the X and Y coordinates of the bottom-left corner of the *i*-th rectangle. xi_2 and yi_2 are the X and Y coordinates of the top-right corner of the *i*-th rectangle.

Output Format

The execution of your program should create a file that consists of two parts. The first part gives the coordinates of color density windows and the densities for color-A and color-B respectively. The second part shows the colors being assigned to the shapes in each individual coloring graph. There is a comma (,) between two consecutive coordinates. Below is the format of an output file.

WIN[d]=x_bottom_left$_d$,y_bottom_left$_d$,x_top_right$_d$,y_top_right$_d$(color_A_density$_d$ color_B_density$_d$)

…

GROUP

NO[i]=x_bottom_left$_i$,y_bottom_left$_i$,x_top_right$_i$,y_top_right$_i$

…

GROUP

CA[a]=x_bottom_left$_a$,y_bottom_left$_a$,x_top_right$_a$,y_top_right$_a$

…

CB[b]=x_bottom_left$_b$,y_bottom_left$_b$,x_top_right$_b$,y_top_right$_b$

…

GROUP

…

WIN[d] indicates the *d*-th color density window for *d* = 1, 2, 3, … . The right hand side of the equality sign gives the coordinates of the underlying color density window. They are then followed by the density values for color-A and color-B.

Behind the information about color density windows is the coloring information of coloring graphs. The information about a coloring graph starts with keyword *GROUP*. It is then followed by some lines starting with either keyword *NO*, *CA*, or *CB*. The order of coloring graphs shown in the output file is immaterial.

NO[i] indicates the *i*-th shapes of a non-2-colorable coloring graph for *i* = 1, 2, 3, … . The right hand side of the equality sign gives the coordinates of the *i*-th shapes. There are no *CA* and *CB* in such a *GROUP*.

CA[a] shows the color assigned to the *a*-th shape for *a* = 1, 2, 3, … .

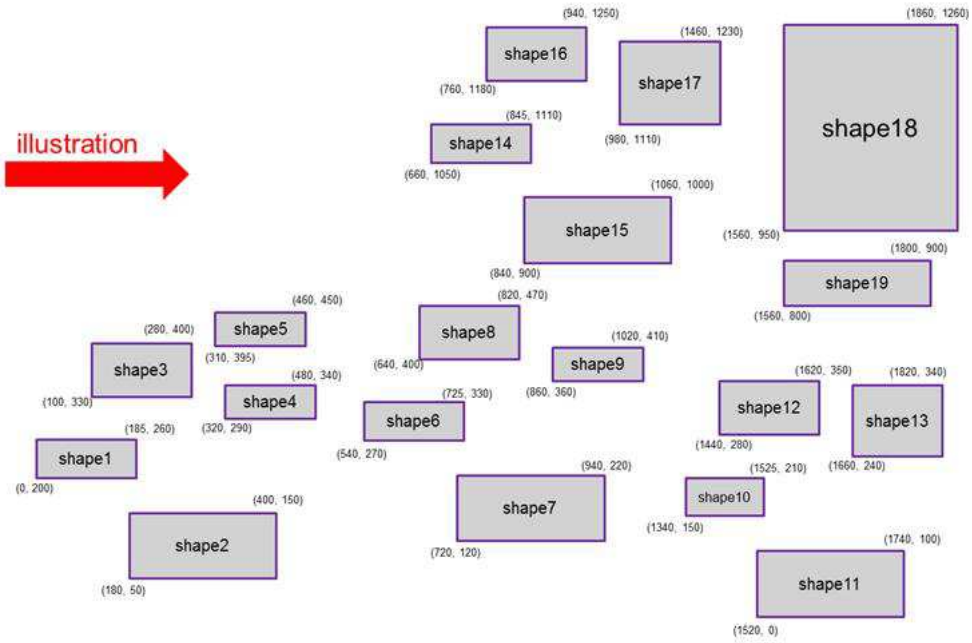
CB[b] shows the color assigned to the *b*-th shape for *b* = 1, 2, 3, … .
Note that the order of rectangles shown in a coloring graph is immaterial.

Examples of Input/Output Files

Input File

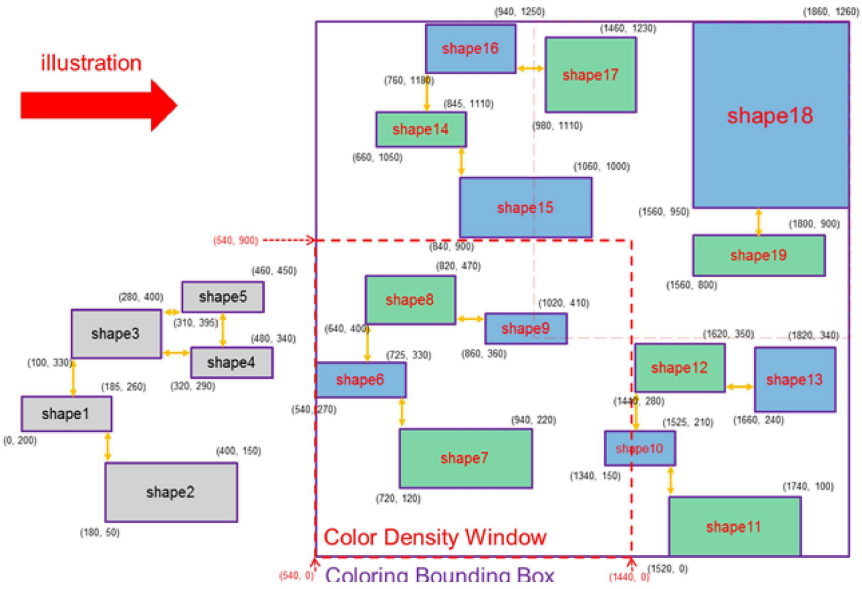
ALPHA=50

BETA=100
OMEGA=900
0,200,185,260
180,50,400,150
100,330,280,400
320,290,480,340
310,395,460,450
540,270,725,330
720,120,940,220
640,400,820,470
860,360,1020,410
1340,150,1525,210
1520,0,1740,100
1440,280,1620,350
1660,240,1820,340
660,1050,845,1110
840,900,1060,1000
760,1180,940,1250
980,1110,1460,1230
1560,950,1860,1260
1560,800,1800,900



Output File

WIN[1]=540,0,1440,900(4.27 3.10)
WIN[2]=960, 0,1860,900(7.23 3.72)
WIN[3]=540,360,1440,1260(9.51 5.26)
WIN[4]=960,360,1860,1260(10.07 13.09)
GROUP
NO[1]=0,200,185,260
NO[2]=180,50,400,150
NO[3]=100,330,280,400
NO[4]=320,290,480,340
NO[5]=310,395,460,450
GROUP
CA[1]=720,120,940,220
CA[2]=640,400,820,470
CB[1]=540,270,725,330
CB[2]=860,360,1020,410
GROUP
CA[1]=1520,0,1740,100
CA[2]=1440,280,1620,350
CB[1]=1340,150,1525,210
CB[2]=1660,240,1820,340
GROUP
CA[1]=660,1050,845,1110
CA[2]=980,1110,1460,1230
CB[1]=840,900,1060,1000
CB[2]=760,1180,940,1250
GROUP
CA[1]=1560,800,1800,900
CB[1]=1560,950,1860,1260



Language

Use C or C++ to do programming. The binary file should be called as DPT_balance_color. Please follow the following format to get the output file.

./DPT_balance_color ($input_file_name) ($input_file_name).out

For example, an input file is named “case1” and the corresponding output file should be “case1.out”. The command line is looked like:

./DPT_balance_color case1 case1.out

Platform

OS: Linux RedHat 4, 5.7 or 5.9

Compiler: GCC 4.5.x, 4.7.x

References

[1] K.M. Monahan, “Enabling Double Patterning at the 32nm Node”, Semiconductor Manufacturing 2006 IEEE International Symposium, pp.126-129, ISBN 978-4-9904138-0-4.

[2] http://www.cs.cornell.edu/Courses/cs3110/2009sp/recitations/rec22.html

[3] Alfred K. Wong. Resolution Enhancement Techniques in Optical Lithography. SPIE Publications, 2001.

Test Cases

1. Several test cases will be provided to you for evaluating your approach.
2. Several test cases are non-disclosure and will not be provided.
3. Click here to Download

Evaluation

The score is 100 for each test case. If your program encounters compilation errors, crash (coredump), runs more than 1 hour or needs more than 4G-byte memory for a test case, the score for this test case will be 0. The final score is the average of the scores of all the test cases.

$$final\_score = \frac{1}{n}\sum_{x=1}^{n} score(x)$$

where score(x) ≥ 0 for x=1, 2, …, n, assuming there are *n* test cases.

For a test case *x*, if your program runs less than or equal to 1 hour and uses less than or equal to 4G-byte memory, we calculate its score as follows.

$$score(x) = f(x) + g(x) + h(x)$$

where

$$f(x) = \begin{cases} 20, & \text{if coloring graphs for test case } x \text{ are built correctly} \\ 0, & \text{others} \end{cases}$$

$$g(x) = \begin{cases} 10, & \text{if color density windows for test case } x \text{ are shown correctly} \\ 0, & \text{others} \end{cases}$$

$$h(x) = \begin{cases} \sum_{d=1}^{k(x)} \left| \frac{70}{k(x)} - \frac{|color\_A\_density_d - color\_B\_density_d|}{5} \right|, & \text{if coloring for test case } x \text{ is correct} \\ 0, & \text{others} \end{cases}$$

$k(x)$ is the number of color density windows in test case *x*. $color\_A\_density_d$ and $color\_B\_density_d$ are respectively color-A density and color-B density in the *d*-th color density window.

For example, if the following output file gives correct coloring density windows and coloring information, its score is (20+10+(70/4-(4.27-3.10)/5)+(70/4-(7.23-3.72)/5)+(70/4-(9.51-5.26)/5)+(70/4-(13.09-10.07)/5)=97.61.

```
WIN[1]=540,0,1440,900(4.27 3.10)
WIN[2]=960, 0,1860,900(7.23 3.72)
WIN[3]=540,360,1440,1260(9.51 5.26)
WIN[4]=960,360,1860,1260(10.07 13.09)
GROUP
NO[1]=0,200,185,260
NO[2]=180,50,400,150
NO[3]=100,330,280,400
NO[4]=320,290,480,340
NO[5]=310,395,460,450
GROUP
CA[1]=720,120,940,220
CA[2]=640,400,820,470
CB[1]=540,270,725,330
CB[2]=860,360,1020,410
GROUP
CA[1]=1520,0,1740,100
CA[2]=1440,280,1620,350
CB[1]=1340,150,1525,210
CB[2]=1660,240,1820,340
GROUP
CA[1]=660,1050,845,1110
CA[2]=980,1110,1460,1230
CB[1]=840,900,1060,1000
CB[2]=760,1180,940,1250
GROUP
CA[1]=1560,800,1800,900
CB[1]=1560,950,1860,1260
```

If any two teams have the same score, the team using less time is the winner.

If any two teams have the same score and use the same runtime, the team using less memory is the winner.

Alpha Test

- Alpha test top 10
- Beta test Top 10:

iccad2015_input.case1	iccad2015_input.case2	iccad2015_input.case3	iccad2015_input.case4	iccad2015_input.case5	iccad2015_input.case6	iccad20
99.46	95.07	91.28	99.78	99.77	99.76	
99.46	95.07	91.12	99.78	99.69	99.59	
99.46	94.7	90.45	99.78	99.7	99.63	
99.46	95.07	91.34	99.78	99.67	98.93	
99.23	95.07	89.9	99.27	99.57	99.44	
99.46	95.07	89.04	99.78	99.69	99.36	
99.46	94.98	89.18	99.63	99.59	99.33	
99.46	95.07	88.67	99.78	99.66	99.24	
99.46	95.07	87.05	99.78	99.52	99.07	
99.46	95.07	87.86	99.78	99.65	99.29	

Q&A

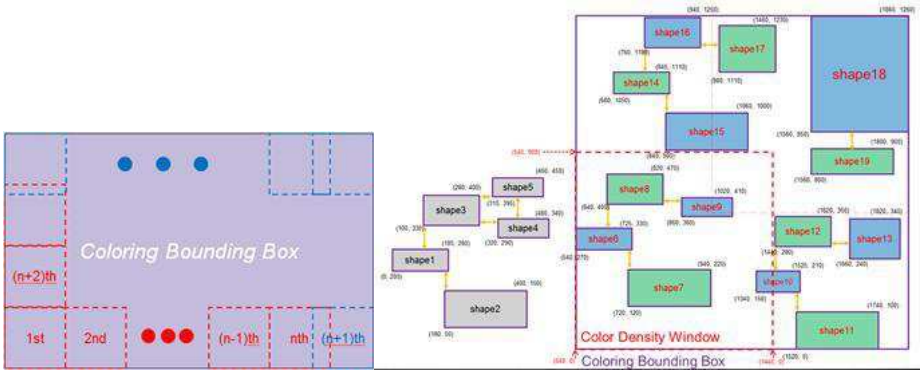
1. 在Examples of Input/Output Files中，
   Output 的shape10剛好被Window分開，W1的GROUP如下:
   CA[1]=720,120,940,220
   CA[2]=640,400,820,470
   CB[1]=540,270,725,330
   CB[2]=860,360,1020,410
   並沒有將被Window切割出的面積計算進去，請問是遺漏了還是正確的結果?

   Output Format

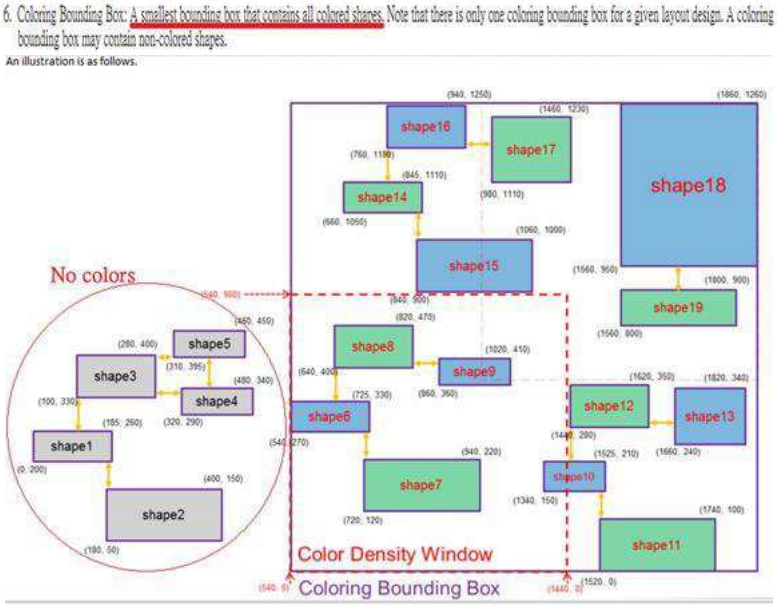
   The execution of your program should create a file that consists of two parts. The first part gives the coordinates of color density windows and the densities for color-A and color-B respectively. The second part shows the colors being assigned to the shapes in each individual coloring graph. There is a comma (,) between two consecutive coordinates. Below is the format of an output file.

   ANS: Window的訊息是放在第一部分, 而第二部分, 每一個GROUP之後接的都是一個完整的coloring graph, 所以這個範例是正確的結果.

2. I have copyed the image from the contest website.The following guidelines 1 writeed that The first color density window is placed at the bottom-left corner of the coloring bounding box.The next image show that window begin with coordinate(540,0) but not (0,0). Is that a error? Or it's just an example that calculate the area and density.Can you explain the problem ?



ANS: Because the coloring graph enclosed by red circle has no colors, the “Coloring Bounding Box” does not enclose shape1, shape2, shape3, shape4 and shape5.
Therefore, the first “Color Density Window” is placed at (540, 0).



3. 關於X-Spacing與Y-Spacing的定義有些問題，如圖所示，兩個shape位置極為相近，但是X方向與y方向彼此都沒有看到對方的邊，依造公告的方法定義，此兩shape並不視為衝突狀況，並不需要著不同色，但在實際狀況來看，此兩shape極為相近，是否該視為衝突狀況，應該以另外方式定義呢?

ANS:真實的情況並不是像X-Spacing與Y-Spacing那麼單純, 而題目之所以選擇用X-Spacing與Y-Spacing來決定如何產生coloring graph, 是因為希望參賽者, 將他們的心力盡量放在如何將window中的density difference降低, 而且越小越好,
因為coloring graph會有跨window的情況, 所以這樣的問題已經夠複雜了, 因此不需要考慮shape間corner-corner spacing的問題.

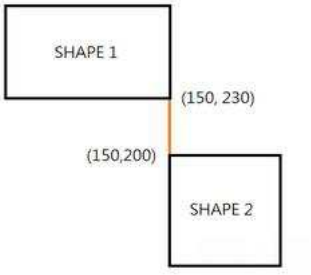
4. 依照題目是：

Given spacing (α and β).
Distance between two edges X(Y) < α(β) must have different color.

倘若A.x2 - B.x1 < α，但是A B的各邊延長不會互相觸及，那麼這樣需要異色嗎？換個說法，若 A 的右下角對齊 B 的左上角，但 A 寬邊延長線不會與B矩形相接，長邊延長線不會與B矩形相接，兩者的 x y 間距皆小於 α β 這樣子需要異色嗎？或者上述只有其一間距條件滿足，需要異色嗎？

ANS:這個題目是不需要考慮corner-corner spacing.因此, 倘若兩個vertical edges其水平方向投影並沒有重疊的線段, 則無須考慮X-Spacing;倘若兩個horizontal edges其垂直方向投影並沒有重疊的線段, 則無須考慮Y-Spacing.

5. 我想請問一下，假如遇到下圖的情形，兩個SHAPE的邊在X軸上的值剛好相同，而Y軸小於Y-Spacing那麼該著成不同的顏色嗎?



ANS: 以你所畫的圖來看, 因為此二水平線段, 其垂直方向的投影, 並沒有重疊的線段, 所以不需要考慮Y-Spacing, 因此無須著成不同的顏色.

6. shape 6 與 shape 10 各別在group中最左側且被歸類為顏色CB，shape 14 在它的group中也是一樣在最左側,卻被歸類為CA，請問是什麼原因呢?
ANS: 在這個例子中, 綠色代表CA, 藍色代表CB, 而此例主要是告訴讀者, 當已經著色著好的情況下, 如何去計算density. 至於該如何著色, 則是大家需要認真思考的地方.

7. 關於訂題組E題的Output File中敘述，
WIN[1]=540,0,1440,900(4.27 3.10)
WIN[2]=960, 0,1860,900(7.23 3.72)
WIN[3]=540,360,1440,1260(9.51 5.26) WIN[4]=960,360,1860,1260(10.07 13.09)
其中WIN[3]=540,360,1440,1260(9.51 5.26)的9.51是不是有算錯?
是否該更正為9.74呢?
我對照比賽的資訊，過成是否應該為(820-640)(70)+(845-660)(60)+(1440-980)(120) = 78900
而density為(78900/810000)*100%=9.74%
ANS: Yes. The correct density is 9.74%.

8. 請問比賽可以接受寫好的ILP package嗎?我用的是cplex，有最新消息麻煩回覆給我，謝謝。
ANS: 針對本題(E)而言, 請勿使用ILP package/cplex.

9. 針對本題(E)而言, 請勿使用ILP package/cplex ，不曉得有沒有包含禁止使用GNU licensed 的 open source solver呢?
ANS: 請勿使用Open Source Solver.

10. 關於 Evaluation 的部分, h(x) 的計算是否都是取絕對值? 因為後面 density 的範圍可能是在 [0, 100] 的範圍內, 最差的情況可能導致
70 / k(x) - |color_density A - B| / 5 變成負分, 但是取絕對值後又變成正分, 這樣跟題目希望的 balancing 不太符合, 是否應該將最外層的絕對值改一般括號？
ANS: h(x)的計算, 最外層的絕對值符號, 可以改為一般的括號, 其修改後的算法如下:

$$h(x) = \begin{cases} \sum_{i=1}^{n(x)} \left(\frac{70}{n(x)} - \frac{|densityAi - densityBi|}{5}\right), & \text{if coloring is correct in case } x \text{ and its value} \geq 0 \\ 0, & \text{others} \end{cases}$$

11. 想請問如果color bounding box的x方向或y方向比color density window的寬度(omega)還要小的話，那color desity window的長寬都維持omega嗎？
還是應該怎麼處理呢？
ANS: 倘若有這種狀況, 那麼color density window的長寬還是需維持omega.

12. 想請問如果所有的shape最後的結果都不需要著色，那這樣代表沒有coloring bounding box，也就不會有color desity window，這種情況下只需印出group(NO)裡面的所有shape嗎？還是完全不需要考慮這種情況？
ANS: 這一種極端的狀況, 並不會出現在測試資料中, 因它並不是此次命題的重點, 請不用考慮這一種情況。

13. Output format 的group有要按照甚麼順序列出嗎? 還有group內的shape是按照甚麼順序列出的呢?
ANS: 如Examples of Input/Output Files中所描述的, 假如有non-2-colorable的coloring graphs, 請先列出, 之後再放coloring好的coloring graphs.至於GROUP內的shapes無須排序.。

14. 已了解 group內的shape無須排序，那假如同樣都是coloring group的話output要排序嗎？
ANS: 無須排序, 但是請記得一個GROUP只能放一個coloring graph的資訊, 如Output Format中所描述的.

15. 你好，我發現二進位執行檔下載到其他工作站跑的話就無法執行，是否要在readme.txt裡多加一行指令 ./chmod 755 DPT_balance_color ?
ANS: 在alpha test時, 我們會將各位的程式在CIC的machines上make然後執行, 所以, 請先確定各位的程式在CIC的machines上是可以make過, 而且可以執行.

16. E組推廣題的平台是寫OS: Linux RedHat 4, 5.7 or 5.9，Compiler: GCC 4.5.x, 4.7.x，可是測試伺服的版本(gcc --version, g++ --version)，gcc (GCC) 4.1.2 20080704 (Red Hat 4.1.2-55)，這樣的執行檔可否在繳交平台執行？
ANS: In addition to RedHat 4, 5.7 or 5.9, the configuration of CIC is also acceptable. Please be informed CIC upgraded the GCC/G++ version recently.

17. 請問alpha submission是只要提交 Readme 和 執行檔，不需要原始碼？
ANS: The source code is optional in alpha submission. You are encouraged to submit an executable binary along with your source code in alpha submission. Please note the source code is required in beta and final submissions and a program without source code won't be rated.

18. 請問alpha submission是只要提交 Readme 和 執行檔，不需要原始碼？
ANS: The source code is optional in alpha submission. You are encouraged to submit an executable binary along with your source code in alpha submission. Please note the source code is required in beta and final submissions and a program without source code won't be rated.

19. 請問會有shape覆蓋整個window的情況嗎? ex.某一color的density=100
ANS: 在所有的test cases中, 並無此種情況

20. 關於之前的 alpha test ，我們在上傳執行檔時沒有考慮到平台的問題，後來看了 FAQ 才想到我們的程式有可能會沒辦法執行，不知道能不能重新上傳或是有其他的解決方法？
ANS: 對於 alpha test，我們決定無須重傳, 說明如下:
    1. 同學尚可利用 beta test 及 final test 進行繳交測試
    2. 在 alpha test上傳時有附原始碼的同學, 我們會幫他們編譯後再執行
    3. 在 alpha test上傳時只有附執行檔的同學, 我們會嘗試在Red Hat 4的機器上執行。提醒同學, final test 進行最後的 ranking 時, 必須要有附原始碼且可在大會平台執行才能計分.

所以, 提醒同學, 下一次要上傳前請先確認以下幾件事：

(1) 原始碼可否在CIC的機器上編譯過, 產生的執行檔可否執行
(2) 執行檔名是否為DPT_balance_color, run一下是否可產生output file. (./DPT_balance_color ../../Cases_in/iccad2015_input.case1 iccad2015_input.case1.out)

Alpha Test是用人工的方式, 盡量幫同學產生output files；Beta Test與Final Test我們會用script的方式, 幫同學產生output files, 因此倘若同學沒有符合規定, 即使他們所設計的演算法是最好的, 也無法拿到分數, 謝謝.

21. Do I need to submit the source code?.
ANS: Yes. Please note the source code is required in beta and final submissions and a program without source code won't be rated.

22. 請問在解問題可否使用multi-thread的技術，如linux環境下使用pthread library？
ANS:我們不希望參賽者使用multi-thread的技術, 而是希望參賽者將心力放在演算法的設計以及分析.

23. 請問是否不能使用thread或fork的方式來加快速度呢？
ANS:請不要使用thread or fork來加快速度.

24. If I did not submit files in alpha test, may I still submit files for beta test?
ANS: Yes. You are encouraged to submit files for beta test and final test.

25. 想請問測試資料中的color density windows的數量最多會有幾個呢？
ANS: 不會超過一千個。

26. output format 的一個 graph 內的 CA, CB 的順序有規定一定要 CA 在前、CB 在後嗎？
ANS: 可以不用。

27. 所有的座標、面積，是否會超過 INT_MAX (2^31 - 1)？
ANS: 不會。

28. 任一個 shape 的長或寬是否會超過 OMEGA，或是有其餘限制嗎？
ANS: 有可能。

29. OMEGA 是否會大於 bounding box 的長或寬？
ANS: 測試資料中不會有此種情況。

30. 在網頁上有提到算分是score(x) = f(x) + g(x) + h(x)
ANS: 只要f(x)和g(x)有其中一個算法拿0分，是不是h(x)也是算0分?
f(x)與h(x)有關係, 倘若f(x)=0則h(x)=0.
g(x)為獨立計算, 不會受到f(x)與h(x)的影響.

31. (1)請問在兩次測試中的 case 6與 case 7 是否之後會提供?
(2)請問在final test的case是否會重複使用alpha test 和 beta test當中的case?
ANS: (1)Case6與Case7並不會公開.
(2)Final Test中的cases與Alpha Test和Beta Test中所使用的cases一樣.